DOCUMENT RESUME

ED 285 131 CS 008 885

AUTHOR Balajthy, Ernest
TITLE Design and Construction of Computer-Assisted Instructional Material: A Handbook for Reading/Language Arts Teachers.
PUB DATE 87
NOTE 66p.
PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052)

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Computer Software; *Computer Uses in Education; Educational Technology; Elementary Secondary Education; *Language Arts; *Reading Instruction
IDENTIFIERS Apple (Computer); BASIC Programing Language; *Software Development

ABSTRACT

Intended for reading and language arts teachers at all educational levels, this guide presents information to be used by teachers in constructing their own computer assisted educational software using the BASIC programming language and Apple computers. Part 1 provides an overview of the components of traditional tutorial and drill-and-practice computer assisted instructional (CAI) software. A history of CAI design follows, beginning with linear programs, and proceeding to branching programs, frame protocols (information frames, question frames, and prompts), and types of feedback. Sources of information for the history section and a list of commercial software are included. Part 2 outlines the basics of planning and constructing CAI software in nine steps. Step 1 involves unit plans, with substeps including choice of topic, goal statement formulation, and topic research. Lesson plans are discussed in Step 2, and the rough draft stage is dealt with in Step 3. Screen Grids, where the CAI author begins to transfer contents of the draft to the screen, are exemplified in Step 4, while Step 5 covers the program rough draft. Steps 6 through 8 focus on the program's different pages, and Step 9 shows how to finish and link the screen displays. A section describing sources of information, including software and books, follows the text. The remainder of the document consists of 14 figures and diagrams that illustrate screen design and provide sample BASIC programs. (SKC)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED285131

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

X This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Design and Construction of Computer-Assisted Instructional Material: A Handbook for Reading/Language Arts Teachers

Ernest Balajthy
State University of New York at Geneseo

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Ernest Balajthy

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

CS 008 885

PART ONE

DESIGN AND CONSTRUCTION OF COMPUTER-ASSISTED INSTRUCTIONAL SOFTWARE

The purpose of this section is to give an overview of the components of traditional tutorial and drill-and-practice computer-assisted instructional software. Such information will be of help in consideration of the "how-to-do-it" section on construction of software that follows. It will also help in evaluation of commercial software, giving guidelines as to what features characterize well-planned programs.

Traditional computer-assisted instruction is much-maligned within contemporary educational circles. This section will briefly discuss some of the dominant criticisms of tutorial CAI, arguing that, despite its limitations, CAI tutorials can be designed and used effectively. The major purpose of this section, however, is to present the design components of CAI tutorials. How is information to be presented in a manner that encourages learning? What matters of concern must a CAI developer bear in mind?

The section begins with a brief overview and history of CAI design. Since this history is closely interwoven with the design of programmed instruction, the instructional principles involved in that area are discussed. Then the major components of a typical CAI program are listed. Special attention is paid to the manner in which information is presented to learners and to the questioning activities that are so central to programmed

instruction.

Typical Criticisms of Traditional Computer-Assisted Instruction

CAI is Dull

In the course of writing this handbook, I received a small grant to investigate integrated learning systems, the large-scale CAI commercial ventures. As I visited school after school, teachers indicated pleasure that their students had access to computer-based instruction. Many teachers noted, however, usually with some puzzlement, that students disliked the time they spent with the computers and had to be strictly supervised to maintain attention.

One teacher paid a backhanded compliment to her school's learning system: "My students prefer the computers to doing workbook drills." Another teacher suggested that behavior problems during computer-based instruction could be handled by constantly patrolling the computer laboratory, monitoring and encouraging students.

While these integrated learning systems offer many advantages over microcomputer-based instruction, it took no great power of discernment to recognize that developers paid minimal attention to maintaining student interest. The typical lesson was composed of dull text drills. The text presentations were uninspired--flat prose cranked out by writers who were interested solely in quantity and who made no efforts to give their writings

personality or style.

Teachers in schools with integrated learning systems were invariably also using microcomputers in their classrooms. Each teacher rated the microcomputer software as highly motivating. This growing use of microcomputers in education has placed great pressures on integrated learning system publishers. Microcomputer software is developed in small chunks, individual programs of limited scope. This discourages integration, but developers are more likely to put a respectable effort into producing and refining each program. Increased competition among microcomputer software publishers has also led to greater efforts in program development. The increasing variety of quality microcomputer programs has shown that computer-based education can be motivational if based on appropriate instructional design.

Teachers Should Not Be Expected to Design Their Own Software

Should teachers construct their own software? From one perspective, this question is as valid as, "Should teachers construct their own lesson plans?" Why not simply follow the directions from the textbook's teacher manual? After all, experienced instructional designers, educators and writers, have joined forces to construct these lesson plans. Why shouldn't teachers follow the commercially prepared materials?

The answer cuts right to the heart of the teaching/learning process. Is teaching a science in the way that chemistry or

physics are sciences? The field of instructional design, whether in basal reading series or in computer-based instruction, is based on the premise that designers removed from the classroom context have the responsibility for planning instruction.

But who can best construct a program that fits into the intricate interaction between an individual teacher, a unique set of students, and a curriculum? The suggestion that editors and writers half a continent away can deal with this interaction cannot be taken seriously. Commercial materials are helpful, beyond any doubt, but they do not meet every need.

Theorists removed from the scene cannot predict learning reactions as simply as they might predict chemical reactions. Each teacher must instead determine what is best for his or her students in each particular situation.

There is no doubt but that individual teachers cannot be expected to design all the software used in their classrooms, any more than teachers can design all their non-computer lessons and materials. There is also little doubt that most teachers cannot afford the time necessary to become accomplished programmers in languages such as BASIC or Pascal. As computers become a greater part of everyday classroom instruction, however, many teachers will find the CAI authoring systems such as PILOT simple to learn and use.

The key problem, however, will not be in teacher training for programming or in provision of teacher time for CAI lesson construction. School systems have it within their power to solve both these problems if the need is felt seriously. The difficult problem will be to produce teacher-developed computer programs

that teach "a real chunk of knowledge or skills, and which don't bore your students into distraction" (Morris, 1984, p. 13). Schools need software that is effective and imaginative.

The Computer Should Be Used as a Tool Rather Than as a Teacher

Educational circles are divided between those who advocate an emphasis on direct instruction and those who advocate an emphasis on learning by doing and by discovery. The debate is sometimes referred to as product versus process, or teacher-centered versus child-centered. Within the fields of reading and language arts, this debate has focused on the subskill ("Children learn language skills by studying the components of those skills.") versus holistic ("Children learn to read by reading and learn to write by writing.") controversy (see Balajthy, 1986, Chapters 7 and 8 for a discussion of this debate in terms of computer instruction).

Despite the increased integration of computers in classrooms, many educational theorists have remained unconvinced that this technology can play an important role in direct instruction. Instead, they suggest that teachers use their computers for more "imaginative" applications such as a tool for writing (as in word processing) or research (as in database management) or as a simulation device.

This skepticism about direct instruction via computers arises from a variety of sources. First, it is well-known that

attempts to replace teachers with computers have had negative results. Second, few have recognized the valuable function of computers as adjuncts to teachers and to traditional materials such as basal readers rather than as replacements. Third, the limited availability of microcomputers in classrooms during the early years of this technology has not lent itself to ease in management of instruction.

Fourth, market demands have led to a wide availability of simple drill software, but little tutorial software is presently available for computers. Fifth, the fastest growing software developers have been independent publishers providing for supplemental instructional tasks. Teachers have had difficulty integrating the great variety of programs into their existing curricula. Sixth, and perhaps most important, there is no doubt but that use of computers as "tools" will be an important function of this technology. No one doubts that word processing is a revolutionary application of computers in the teaching of writing, for example.

Yet, while the problems must be acknowledged, these criticisms do not overcome the promise of computer-assisted instruction for improvement of education. There is an increasing recognition of the importance of well-organized direct instruction that emphasizes explicit teaching of concepts and skills. It is true that the holistic movements within the field of reading, such as Individualized Reading, Language Experience Approach, and Shared Books, have played a vital role in increasing educators' awareness of the need for providing children with rich language experiences. Yet the rapidly

increasing research on the dramatic effectiveness of direct instruction clearly indicates the central role it must play.

The History of CAI Development

The history of CAI development is a story of increasing sophistication in design of courseware. This sophistication has been reflected in a steady movement away from CAI's foundations in rigid, computer-centered (by which we mean, curriculum-centered, as the computer simply presents the curriculum) methods toward approaches that are learner-centered, more sensitive to the students' needs.

This section is not a "history," per se. Instead, it deals more with the realizations by CAI designers of better ways to present learning materials through use of computers: How those realizations were made and what results came of the realizations. The central events had to do with the beginnings of non-electronic programmed instruction in rigid, linear formats. The next major movement was toward the increased flexibility of computer response to the learner involved in branching programs. A new, major step toward even greater sophistication in learner-centered CAI is taking place in ICAI (Intelligent Computer-Assisted Instruction).

Linear Programs

The development of linear approaches to CAI rests on the findings of behavioral psychology, the school of thought that dominated educational psychology for the first three-quarters of the twentieth century. Among its leaders were E. L. Thorndike and B. F. Skinner.

Behavioral psychology is usually thought of as having its foundations in Thorndike's (1898) "Law of Effect." In working with animals, Thorndike demonstrated that behaviors that are followed by pleasurable experiences are more likely to be repeated than those that are not.

Skinner elaborated on Thorndike's ideas to further develop this "stimulus-response" (S-R) theory. In his examination of behavioral applications in education, The Technology of Teaching, he suggested that education involves the results of an appropriate arrangement of what could be considered pleasurable sensations (technically defined and refined as reinforcements) experienced in conjunction with learning: Teaching is "simply the arrangement of contingencies of reinforcement" (1968, p. 5).

Two basic principles characterized this behavioral approach to learning. One involves the complexity of learning. Since most academic learning (which is the "desired behavior") is complex, it will not occur spontaneously. That is, one cannot simply wait until a child has learned to read critically before one supplies the reinforcement for that activity. Instead, teaching must involve reinforcement of successive approximations to the desired final behavior (a teaching process called shaping

behavior): Teaching must occur in small steps, with appropriate reinforcements at each step. As a result, in traditional programmed instruction and in CAI tutorials, information is presented in small portions, called frames. One frame of information is presented on each screen page. Learning occurs as frame builds upon frame, step by step.

A second basic principle involves the type of reinforcement appropriate for learning. Both programmed instruction and CAI depend upon the reinforcing value of correct answers. Learners feel rewarded when they are able to come up with a correct answer to a question. This reinforcement is all the motivation a student needs, as long as feedback is immediate: "The lapse of only a few seconds between response and reinforcement destroys most of the effect" (Skinner, 1968, p. 16). As a result, in the construction of materials, the questioning must be designed so that students progress bit by bit, finding each question easy to answer, and receiving prompt feedback.

In linear programming, then, programs consist of a sequence of frames, each of which represents a small step toward the desired learning behavior. Students are asked questions about the material immediately, each of which they find easy to answer at their stage in the learning process. Their input actively involves them in learning, and it receives immediate feedback as to its accuracy. The sequence of frames in a linear program is characterized by its unvarying nature. The "line" of learning is the same for every student. Since no incorrect responses are supposed to occur, no individualization is necessary, though

students are sent back to previous material if they do answer incorrectly. Only the rate through which students progress in the program varies.

Branching Programs

Programmed instruction designers soon recognized the problems inherent in the unvarying sequential presentation of material. The assumption that every student learns in the same way is today advocated by virtually no one in education. O'Shea and Self (1983), for example, noted that, "The poverty of linear programming is so manifest that the technique has long been extinct in computer-assisted learning" (p. 71). Contemporary software publishers, many of whom continue to use this linear approach, do so only because of lack of understanding of CAI principles on the part of the program designers or because of inadequate financial backing for program development.

Branching tutorials attempt to provide increased individualization, though in doing so they violate the principles of the more linear, traditional operant learning theory advocated by B. F. Skinner. The branching program author predicts student needs and provides separate "branches" within the program to meet each need. If a student experiences failure, the program provides extra explanations and practice. It may even attempt to diagnose the failings and provide appropriate remediation targeted to the diagnosed problems.

In an early statement of advocacy for branching programmed instruction, Crowder (1959) suggested the need for closer

attention to student responses. "The student's response serves primarily as a means of determining whether the communication process has been effective and at the same time allows appropriate corrective action to be taken when the communication has been ineffective" (p. 114). Branched instruction provides a flexibility of response to learners' needs, though it also greatly increases the responsibility and work of the program author, who must anticipate the various needs and their solutions.

Branching tutorials are similar to linear tutorials in that both present information, require a student response to that information, then provide feedback on that response. Linear tutorials, however, attempt to guarantee correct answers in part by providing information in small chunks. Branching tutorials generally provide more information per frame or more frames of information prior to a question. Correct answers are not absolutely required. In addition, in branching tutorials, students work through the different subpaths of the program as their different needs require.

Frame Protocols

The term frame means the amount of information that is presented to the student at any one time during programmed instruction. In CAI, this term refers specifically to the amount of information provided on the computer screen at a particular time. Other terms that can be used as synonyms are screen,

screen page, and display.

CAI developers make use of a wide variety of frame protocols, sample frames that serve as models for design. The quality of a CAI lesson depends largely on the manner in which these frames are strung together to teach the targeted skill or concept. The developer tries to choose the types of frames most appropriate for each learning event in the program, varying the presentation to help make the program more interesting.

There are two major categories of frames and several minor categories. The major categories are information frames and question frames. The minor categories include such items as title pages for each lesson, menu pages for selection of lessons, pages of directions, and so forth.

Information Frames

The first major category of frame is called an information frame (or teaching frame). Such screen pages simply present information to the student. This information might include facts about the topic, instructions for carrying out a skill, or examples of a skill or concept.

Question Frames

The second major category of frame is called a question frame (or criterion frame). These frames promote student involvement in the learning process by requiring a response to a question posed. Developers attempt to design the program in such

a way that learning becomes a dialogue between the computer and the student. Questions help reinforce learning, as the student can be asked a variety of questions about each topic. Questions also serve the function of testing, probing student learning to determine success or failure.

CAI developers can choose from a variety of question frames. Forced-choice frames are the most common. These require students to choose from a given selection of possibilities, as in a true-false or yes-no frame. Multiple choice questions, in which the student types A, B, C or D in response to a question and its possible answers, are another type of forced-choice frame, as are matching item questions. In such question frames, the first part of the item is called the stem, and the possible answers are called the foils. One of the foils is usually correct, and the remainder are called distractors.

One variety of multiple choice frame is the BABOON frame. BABOON is an acronym for B, A, BO(th), O(r) N(either). These frames require more thought than most multiple choice formats, as the student must consider the possibilities that two foils are correct or that none are. For example,

THE TOMATO IS THEORETICALLY CLASSIFIED AS

A. A FRUIT

B. A DRUPE

C. BOTH A AND B ARE CORRECT.

D. NEITHER A NOR B ARE CORRECT.

CAI developers can also choose to provide answer-supply

questions (also called constructed response frames). These include fill-in-the-blank questions, sentence completion questions, and short answer questions.

Answer-supply questions play a limited role in reading and language arts software construction, as developers must anticipate all possible correct responses. The computer's ability to accept open-ended responses from students is limited. These frame protocols are appropriate only under circumstances in which correct answers can be exactly predicted. For example, a developer might want to use such a frame when a specific technical term is being taught and its use is to be tested, or when students have received prior instruction to answer all questions using a limited choice of words, as in a subject-verb agreement drill on use of was and were.

Developers must be wary of the temptation to force CAI open-ended frames to do things they were not designed to do. They must also not mistake the simple ability to parrot a technical term for a true understanding of the principles involved. A possibly apocryphal story of John Dewey illustrates how simple-minded question answering does not always indicate true understanding:

Dewey was visiting a classroom in which youngsters were studying geology. He asked the students to tell him, "What would happen if you dug a deep hole in the earth?" None of the students was able to give him the expected answer, having to do with the earth's molten core.

The teacher broke into Dewey's discussion, noting that he

had asked the "wrong" question. "What is the state of the center of the earth?" she asked the students.

In unison, they responded, "Igneous fusion." They knew the technical term, but were not able to explain its meaning out of the context of the exact question wording. The teacher had not realized that if students cannot transfer their learning, true learning has not taken place. CAI designers must be aware of the same problem, and they should provide opportunities for use of the student learning in a wide variety of contexts.

Another questioning problem involves inexact questioning. Developers, for example, should phrase questions so that students understand exactly how to respond. For example,

The red airplane flew in circles around the crowd,
slowly coming closer and closer to the ground.

Do you know the color of the airplane?

_____

Unless the program author expects a yes-no response, the question is poorly phrased.

Prompts

As noted earlier, the complexity of most school learning requires that behaviors be shaped. That is, teaching must be targeted to obtaining responses that are successively closer to the desired final responses. A program must be designed to avoid

posing questions that are so complex as to bog students down. One method of shaping responses involves the provision of prompts, hints as to the correct answer.

Developers can use a variety of prompt types. The aim in each case is to provide enough of a hint so that students can answer the question, but not so much that students find the question so easy that no thinking is involved.

One common prompt type involves provision of the initial letter of the desired response. If word endings may cause confusion, as in the use of plurals or past tense, the final letters may be provided. Another type of prompt involves giving students the number of spaces that match the number of letters or number of words in the desired response.

Some programs allow variable prompting in which the student chooses the amount of prompting necessary. This technique is frequent in word games. The more clues required, the fewer points won when the word is guessed. In the vocabulary game Quizit, for example, students are presented a definition and a contextual sentence with the appropriate number of spaces for letters in the word to be guessed (see Figure 1). They can then request as many initial letters as they think they need to guess the word. The fewer letters requested, the more points won when the word is finally guessed.

Still other programs provide for a "clue" or "hint" option that may be requested by the student. In a reading comprehension program, for instance, the appropriate sentences in the target paragraph might be highlighted to help students answer the

question, or perhaps some additional hints as to the answer might be given.

Figure 1. Screen page from Quizit

```
BECOME ADJUSTED

THE LENS OF THE EYE WILL ^^^^^^^^^^
ITS SHAPE FOR DIFFERENT DISTANCES.

----------------------------------------
WELL, THIS IS TOUGH, PAUL.

THIS WORD HAS 11 LETTERS.

HOW MANY DO YOU WANT (0 TO 11)?
----------------------------------------
```

Types of Feedback

The function of feedback in CAI is often poorly understood by software evaluators and purchasers. Part of the misunderstanding results from the difference between feedback for reinforcement and feedback for development of automaticity.

When students are first being taught a skill or concept, their learning requires reinforcement. Questions are presented to the students to help them learn the material presented accurately. Feedback for such reinforcement questions must be detailed and specific. Response-specific feedback requires the program developer to anticipate many possible responses on the part of the student, both accurate and inaccurate. For each type of inaccurate response, a different type of feedback is provided to better explain where the students' reasoning went awry.

Once students have learned the skill or concept, they often need additional practice to "overlearn" the material so as to reduce forgetting or to speed performance or to develop automaticity of performance (that is, to develop performance of the skill without the devotion of attention to that performance). A vocabulary drill, for example, might be designed to speed word recognition so that students need not slow down their reading in order to think of the word when they see it in a passage. Feedback for such learning tasks need not be detailed. Instead, exposure rate is the more important issue. The program would be designed to offer students many examples at a fairly rapid rate of speed with minimal feedback.

Positive feedback. Provision of feedback is one of the major responsibilities of the program developer. Positive feedback may involve telling the students they are correct or may involve a compliment such as "Good work!" or "Nice job, Tom!" Many program designers have found that simply moving on to the next problem is sufficient positive feedback to maintain interest. Others get imaginative in their feedback, providing an ever-changing variety of compliments, a simple tune, a graphics display, or catchy congratulations:

"You're the pick of the carrot patch, Lisa. You got it right!"

"Out of sight--that's so right!"

"Good show. You're a pro!"

(Chan and Korostoff, 1984)

Negative feedback. Dealing with wrong answers offers an

even greater challenge to CAI authors. Questions must be closely analyzed in order to predict possible incorrect responses and to branch to the appropriate remedial frames that can explain why the answer was wrong.

One common form of feedback is to recycle back to the question for one more try, perhaps with the additional help of a hint. In some cases, authors may wish to program the input operation so that incorrect responses will not be accepted by the computer. One good example where this could be helpful would be in spelling instruction, where authors would not want the incorrect spelling of a word to appear on the screen for fear that the errors will be further visually reinforced.

Unanticipated responses. No matter how closely the author analyzes each question, there will be times when students make responses that are completely unanticipated. Some programs respond with a comment to the effect, "I do not understand that answer," then recycle back to the question for another response. Others may automatically conclude that the response is incorrect and respond appropriately. The danger in this latter approach is that students may be offering an answer that is correct in some way. Negative feedback may mislead the students or weaken their confidence in the program.

Null response. In some cases a student will be unable to offer any answer to the question posed. Directions to the students should cover this possibility. One standard method of dealing with such a case is to have students simply press the RETURN key. In other cases, it may be appropriate for the computer to time student response latency and automatically move

on if the student hasn't responded within a reasonable period.

SOURCES OF INFORMATION

SKINNER, B. F. 1968. The Technology of Teaching. New York: Appleton-Century-Crofts.

This text provides a detailed look at Skinner's important work with pre-electronic programmed instruction. Skinner strongly explains and supports his advocacy of programmed tutorials and drills. Besides providing explanations as to what programmed instruction offers the field of education, this text serves as an invaluable overview of historical interest.

WALKER, DECKER F., AND ROBERT D. HESS (eds.). 1984. Instructional Software: Principles and Perspectives for Design and Use. Belmont, Cal.: Wadsworth.

This text contains a collection of essays on various matters of importance to design of educational software, including psychological principles underlying CBI, motivation, provision for student-computer dialogue, and design of simulation software. In addition, there are several articles discussing implications of artificial intelligence for computer-based instruction.

COMMERCIAL SOFTWARE

Quizit

Regents/ALA

Two Park Avenue

New York, NY 10016

Apple II-series

PART TWO

PLANNING AND CONSTRUCTING CAI SOFTWARE:

THE BASICS

The purpose of this section is to introduce the basic format of a traditional computer-assisted instructional unit in a reading skill. Many computer-using teachers have had courses in programming BASIC or Pascal. Often, however, these courses have never covered the topic of designing educational CAI tutorials. Instead, they concentrated on business applications of programming. This section presents the format of constructing a simple tutorial lesson in a simplified form which should be understandable by most teachers, whether or not they have formally studied programming.

We will start with a consideration of our goals and objectives. Curriculum content and methods are considered next. Both these preliminary topics deal more with the field of curriculum design than with computer-based instruction, but a carefully considered curricular foundation must be laid right from the start. Simple computerization of teaching material does not automatically grant educational justification to ill-considered content or methodology.

From this foundation we will move to design and construction of the actual computer program. The unit to be partially designed here will be a simple "page-turning" program in which the computer does little more than display pages of text and allow the user to move from page to page. As such, it will be of

no more value than a hard-copy printed textbook. However, the unit can ultimately serve as the skeleton for a finished product, a product which will take advantage of the computer's potential more fully by incorporation of question frames for increased interaction, sound, graphics, and a management system,.

The design and construction of the unit in this section will include laying out text screen grids. There will also be a brief explanation as to how to program the actual pages, and how to link the lessons together into a smoothly running program.

As a concrete example to illustrate the design and construction of a CAI program, we will be using a unit on use of verbal context in determining meaning of unfamiliar words. Since this unit is meant only as an example, it will be demonstrated in short parts--far too incomplete for actual use in a classroom but hopefully sufficient for its explanatory purposes.

For readers familiar with BASIC, lines of programming will be offered. However, for those readers who are unfamiliar with BASIC, all programming will be explained in the text. The actual programming is in Applesoft BASIC, the standard form of the language used in the Apple II-series microcomputers.

As we begin, please bear in mind an important point: For the pedagogical purposes of this text, we are beginning our discussion of a computer instructional unit as if we have no plans for future modifications and improvements--for example, a teacher management system and pre- and posttesting which could be added later on. The fact of the matter is that preplanning of a computer project should be comprehensive. Time spent planning

before programming will save enormous amounts of time and confusion later on. CAI projects should be completely thought out before programming is begun.

Due to the nature of this CAI construction section as a "how-to-do-it" approach to the topic, we have adopted a somewhat informal tone. Hopefully this will add to the ease of understanding of some concepts which are rather complex and intimidating for nonprogrammers.

Note that the design and construction of educational software is a highly complex field of specialization, one which can only be surveyed in this handbook. Our purpose in this text is to "demystify" the CAI design process. That is, we teachers ought to be familiar with the basic procedures used to design and construct CAI software, even if our knowledge is not sufficiently in-depth to do it ourselves. This section serves as an introduction to the topic which will give teachers a clearer picture as to how it is done, and it will also help them to be more informed and critical when they evaluate CAI software.

Step 1: Unit Plans

Choose Your Topic

Perhaps the hardest part of beginning a CAI program for most teachers is deciding on the topic. All programmers realize the enormous amounts of time required to develop a program. They will be married to the program for a long time to come, and like those people considering actual marriage, they want to choose the

"right partner."

In general, three suggestions come to mind. First, choose a skill or content topic which is needed. Considering the dearth of direct instructional CAI material in reading available as of yet for computers, this criterion really doesn't narrow the field by much.

Second, choose a topic and a grade level about which you know something. You don't necessarily have to be the world's leading expert on it--after all, we'll be doing research in preparation for designing the content--but it certainly helps if you've had experience with the topic. Are you a middle grade reading teacher? Choose something having to do with reading in the middle grades. Have you put a lot of effort into social studies with your third graders? Choose a third grade social studies topic.

Third, choose a topic in which you are interested. Your enthusiasm will be a key factor in the quality of your results.

Write Your Goal Statement

Once you've chosen the general topic, develop a goal statement--a single sentence explanation of just what it is you wish your students to learn in this unit. Almost every series of lessons has a host of goals. You want to choose the single most important goal. The topic for our sample unit is a word recognition skill, that of context analysis. Our goal statement is clear and simple, though not very specific (indeed, goal

statements are supposed to be general--we'll get specific a little later on):

* TOPIC: Context Analysis

* GOAL STATEMENT: Students will learn to recognize and use different types of context clues to determine meanings of unfamiliar words.

Notice that the goal statement indicates broad direction and general purpose. It is not concerned with specifiable amounts of learning or time periods. Other examples of possible goals include:

* Students will learn how to use an encyclopedia for independent research.

* Students will learn to successfully add two digit numbers.

* Students will understand the causes and results of the Viet Nam War.

* Students will develop skills in reading, writing, listening, and speaking.

Let's look for a moment at that last goal: "Students will develop skills in reading, writing, listening, and speaking." Very impressive. Covers just about everything, doesn't it? Well, that's the problem. It is indeed a goal statement, but it errs in that it is far too general, too broad. There is no

specificity whatsoever, and a unit based on those goals could include almost anything. In constructing our goal statements, we want to walk the line between too much specificity on the one hand, and vagueness and ambiguity on the other.

Research Your Topic

You may find that your topic is so familiar to you that you can skip this step. That is, you may have already researched the topic quite thoroughly. Most teachers, however, will feel more comfortable with their topics if they do a little research on them. This research will provide both content material and knowledge of instructional methodology. We want to learn more about our topic and about how that topic is taught. Where can we look?

Reference books. Use the library or your college texts as resources. Our example topic, contextual analysis, is drawn from the field of reading. A logical place to start our search for information, then, is in reading education textbooks. A quick look at the index of each book will direct us to the appropriate pages for theory, research, and methodology on contextual analysis.

Professional journals. Up-to-the-minute developments and original ideas are often best researched in journals and magazines. Several journals and magazines are specially designed for reading teachers, and many educational publications periodically offer ideas for the teaching of reading. By scanning the annual index to each, or by looking in the Education

Index or Comprehensive Index to Journals in Education, we can quickly locate articles with likely titles to help us learn about context clues.

Instructional materials. Our final step in the research phase is to look at actual materials designed to help students learn about our topic or skill. Go to the textbooks and workbooks--as well as their teacher manuals--for concrete examples of how the educational publishing establishment attempts to deal with the issue.

Step Two: Lesson Plans

In our research on contextual analysis, we have found that students can use many different strategies to determine the meaning of unfamiliar words from their verbal surroundings. Indeed, there are so many strategies that we must pick and choose, following the general philosophy that it is better to teach a few things well than to try to cover everything and end up with students learning nothing. We will choose three context strategies: Use of synonyms, antonyms, and author-supplied definitions. In addition, we decide that the program should include a general introduction to context strategies as well as a lesson on the limitations of context clues. In all, we will have five lessons (see Figures 2 and 3 at end of section).

Specific Objectives

In order to begin planning the construction of the five lessons, a specific objective must be developed for each. That is, we should make a precise statement as to just what it is the student will learn in that lesson. These statements will carry our planning one more step further along toward specificity.

The curriculum designer may choose from two types of specific objectives. One type is called a performance or behavioral objective. Such objectives are based on behavioral psychology, an "instructional management" approach to education. They are most applicable to teaching of lower level skills, which lend themselves to quantification and measurement somewhat better than more complex skills or ideas. Some examples:

The student will recognize and sound the long and short vowels with 80% accuracy, as measured by the school-wide Test of Basic Skills.

The student will complete ten problems requiring addition of two two-digit numbers in ten minutes with 90% accuracy.

Many teachers prefer the second type of specific objective, called an experience objective. Rather than attempting to quantify student behavior--a process which may tend to mechanize and dehumanize education--this type of objective describes the experience to be given to the child in the lesson. The five experience objectives developed for the Context Analysis Program are:

Objective One: The student will learn about the role of context analysis in relation to other word attack skills (phonics, structural analysis, and dictionary use).
Objective Two: The student will examine examples of context clues illustrating the differing specificity of verbal contexts.
Objective Three: The student will learn that words of similar meaning (synonyms) can be context clues.
Objective Four: The student will learn that words of opposite meaning (antonyms) can be context clues.
Objective Five: The student will learn several ways that authors supply definitions of unfamiliar words.

See Figure 4 for a blank objective plan form.

Lesson Summaries

Most teachers will recognize this next component of the program planning process as the "lesson plan," a short paragraph written to briefly describe three components of the lesson: 1) the content, 2) the sequence, and 3) the methods. That is, the lesson summary will include the highlights of what will be taught, in what order it will be taught, and how it will be taught. If it would be helpful to improve clarity, an example of the skill may be included.

You may feel somewhat discouraged at the time required for this type of planning. Bear in mind that, as time goes by and

you become more accomplished in the art and science of writing curriculum materials, these steps may become superfluous. For now, however, they are vital in that they require you to think out your work as you go along. You might be cheered to know that these plans also have a directly useful application to your programming project: They will be placed in the written documentation which will accompany your program, to help users understand the rationale and operation of the program. In addition, if you are designing software for publication, they will be used in the Program Description you submit to software publishers whom you wish to interest in publishing your program.

Figure 5 presents a completed Lesson Summary Plan Sheet for our Context Analysis Program. Figure 6 is a blank plan sheet for your use.

Step Three: Rough Draft

The purpose of the rough draft is to get your instructional statements, examples, and practice materials down on paper. You will write out the entire contents of each lesson. As you become more experienced in construction of software, this step may be combined with Step Four--Screen Grids. In Step Four you will transfer the contents of your rough draft to form screen grid layouts, hard-copy equivalents of what will actually appear on the monitor screen. With some experience, you will learn to do your rough draft right on the screen grid paper. As a matter of fact, many programmers complete neither rough drafts nor screen grid diagrams. They put their ideas directly into BASIC.

Sometimes this shortcut works. At other times the result is regrettable. At any rate, for novice programmers it is certainly best to work one step at a time.

Take the paper for your rough draft and divide it into two vertical sections, the left section taking up about two-thirds of the sheet. Your contents will be written on the left. Programming suggestions will be included on the right.

Consider constructing your draft as if you are the writer and someone else will be doing the programming. Weeks may go by before you get to do the actual programming of certain sections of your draft. By that time, you may well have forgotten some of your ideas. By writing them down, you'll not have to worry about forgetting just what it was you wanted to do. Landa (1984) gives detailed suggestions as to how to note various programming ideas in a rough draft. Her book is written from the perspective of a non-programming educational materials writer communicating to a programmer.

The computerized final product will only be as good as your rough draft, so devote an appropriate amount of attention to your writing at this stage. Develop a personal checklist of components you wish to include in each lesson. Without going into great detail at this point, it might be suggested that the following components be included, as befits a model based on direct instructional principles (Rosenshine and Stevens, 1986):

1. Purpose-setting statements. Your students should know exactly what it is they are going to be doing, and why they are going to be doing it.

2. Introductory material. An interest-arousing introduction should motivate students. In addition, the topic of the lesson should be related to other topics studied by the students to better enable the connection between existing knowledge and new learning.

3. Instructional statements and questions. This is the body of your lesson, the actual teaching of the skill or content material. Since the computer is an interactive device, encourage interaction by asking for many student responses.

4. Examples. Give examples to develop your students' understanding of the verbal instructional material. Again, requiring student response to the examples can encourage thinking and attention.

5. Practice. In order to enhance long-term retention and fluency of performance, some limited practice (i.e., reinforcement) of the skills or concepts ought to be presented. This is usually not the place for drill. Since the purpose of reinforcement exercises is to develop accuracy (as opposed to the automaticity developed by drillwork), response-specific feedback in which detailed explanations of incorrect answers are provided, should be given.

The actual method of notation is less important than consistency. You must remember just what you wanted. In large-scale curriculum development projects involving more than one person, the system of notation becomes far more important. Everyone must understand and use the same notation system.

Step Four: Screen Grids

Once the rough draft has been completed, the CAI author begins transferring contents of the draft to screen grid diagrams. These grids are laid out to match the spacing available on the text screen. For the Apple II-series, the text screen allows 24 lines of 40 characters each (see Figure 11). Text material is printed clearly in the appropriate spaces on the grid diagram. Accompanying instructions necessary to the programmer are written in the margins of the grid page.

Figures 7 to 10 represent screen grid diagrams for our context analysis program. This short sequence will eventually become the first few screen pages of our sample Lesson 1. Notice the explanations and directions which accompany the text, printed in the margins. The "Press Space Bar" subroutine is labeled SR 100 for future reference in Screen #1 (Figure 7). Later on, when the same subroutine is called up in Screen #4 (Figure 10), only the code name SR 100 is used, to save time. A delay, in which the computer will wait a bit before proceeding, is noted in Screen #2 (Figure 8).

The question asked in Screen #2 (Figure 8) leads to two possible routes. As noted, if A, B or C is pressed, Screen #3 (Figure 9) will be displayed, then the bottom of Screen #4 when Y or N is pressed.

Step Five: Program Rough Plan

Once the entire program has been drawn up on screen grids, the major tasks involve programming. At this point, the courseware author has an accurate sense of just what the final series of programs will look like. We know the approximate length in screen pages, for example, and we can develop a rough plan of action for the actual program which will enable us to begin programming.

Our page turner unit will consist of five separate lessons. For ease in construction, we will program each of the lessons under a separate file name and link them together with commands to access the disk.

Actual roughing out of the program plan will vary greatly depending on the contents. For our purposes, we will allot the beginning section of each program to subroutines to be used. Lines 1 to 999 of the program ought to be more than sufficient for the few simple routines we will use, such as the SR 100 routine to print the "Press any key" message at the bottom of pages.

From there on, we will allot 100 lines of program "space" to each page. While this is far more than required, the length of the program does not force us to make tighter line numberings, and besides, the numbering in quantities of 100 will make for easier reference to specific pages. Lines 1000-1100 will make up Screen Page #1, for instance, and line 1100-1200 will make up Page #2, and so forth. Since all five lessons follow essentially the same format, we will be able to use the same rough plan for

each.

From this point on, the remaining steps involve your knowledge of programming, rather than that of courseware design. Remember that in this section we are not concerned with user friendliness, with crashproofing the program, or with any of the many other related issues vital to courseware development. We will arrive at a "bare-bones" instructional program, for the simple sake of keeping the task manageable so as to keep attention directed to the major purposes of this section. The "bells and whistles" can be added later on.

Step Six: Program Page One

We begin by labeling the program in its first few lines to aid in future identification.

```
1 REM LESSON NUMBER ONE--INTRODUCTION TO CONTEXT ANALYSIS
2 REM ERNEST BALAJTHY
3 REM VERSION 9/27
4 GOTO 1000
```

Line 3 gives the date this program was updated last, an important fact for later identification of your most up-to-date version of the program. Line 4 starts off the program at line 1000, the beginning of Screen Page #1.

Carry out the programming necessary for Screen Page #1. The programming is actually quite simple, consisting primarily of a series of PRINT statements.

To print the lesson number and title:

```
1000 REM SCREEN PAGE #1
1005 VTAB (6): HTAB (5):HOME
```

Line 1005 positions the cursor at the appropriate space to begin printing.

```
1010 PRINT "LESSON ONE"
1015 PRINT: PRINT
```

Line 1015 inserts the two blank lines between the lesson number and its title. The same effect could have been achieved by using the command VTAB 9.

```
1020 PRINT "      INTRODUCTION TO"
```

Line 1020 has the first part of the title preceded by six spaces (i.e., pushes of the space bar). This will indent the word INTRODUCTION six spaces. The same indentation could have been achieved by preceding the PRINT command with HTAB 7:.

```
1025 PRINT: PRINT "          CONTEXT ANALYSIS"
1030 GOSUB 100
```

Now we have the body of text set up for the first screen page. We must pay attention next to setting up the subroutine called up in line 1030, the "Press Space Bar" routine we call SR 100. We will use a very simple (and, correspondingly, error-prone) routine.

```
1  GOTO 1000
100 REM PRESS SPACE BAR ROUTINE
105 VTAB 21: PRINT " PRESS SPACE BAR TO GO ON."
110 GET A$
115 RETURN
```

While the message will read PRESS SPACE BAR TO GO ON, the press of almost any key on the keyboard will be accepted by the

GET A$ statement and cause the program to proceed.

Step Seven: Test Run of Page One

At this point, run the program by typing RUN (or RUN 1000 to begin the program at line 1000. Starting the run at the appropriate line number will save much time later on, as you won't have to page through the earlier pages to proofread the later pages) at the cursor prompt and pressing the RETURN key. Your Screen Page #1 should appear on the screen. A touch of the SPACE BAR will move the program on, but since there is nothing to follow, the program will end and a cursor appear. Correct any apparent mistakes at this point.

Step Eight: Recycle for Page Two

Start programming Screen Page #2 on line 1100, as per your rough program plan. Proceed in the same fashion as with Screen Page #1.

```
1100 REM SCREEN PAGE #2
1105 VTAB (3): HTAB (4):HOME
1110 PRINT "WHAT DO YOU DO WHEN YOU COME"
1115 PRINT:PRINT "       ACROSS A WORD YOU DON'T"
1120 PRINT:PRINT "       UNDERSTAND?"
1123 FOR WT = 1 TO 7000: NEXT WT: REM THIS INSERTS A DELAY
1125 VTAB (11): HTAB (5): PRINT "A. CRY"
1130 PRINT: HTAB (5): PRINT "B. THROW A TANTRUM"
```

```
1135 PRINT: HTAB (5): PRINT "C. HOLD YOUR BREATH TILL YOU TURN"
1140 HTAB (10): PRINT "BLUE."
1145 PRINT: HTAB (5): PRINT "D. NONE OF ABOVE"
1150 VTAB 21: HTAB (3): INPUT "CHOOSE ONE OF THE ABOVE.  ";A$
```

Step Nine: Finish and Linkage

From here on, you face the simple, though time-consuming, task of laying out the many screen displays needed for your five lessons. Remember to save your work frequently, and make backup copies of the unit on separate disks.

Figure 12 and 13 contain a full printout of our abridged Lessons One and Two. Once you have finished the linear series of pages for each lesson, the five lessons must be linked together to form a sequence. The command line

```
CHR$(4); PRINT "RUN LESSON TWO"
```

placed at the end of Lesson One will cause the program entitled LESSON TWO to be accessed from disk and run. Place similar commands at the end of Lessons Two, Three, and Four to link all five programs together in sequence.

To repeat the warning given above, the programs presented in this section are only meant to be introductory examples of the kinds of tasks necessary for construction of a page-turner unit. I hope that the explanation has helped de-mystify the lesson construction process. I also hope that the explanation has helped highlight the real challenge involved in computer-assisted instructional design, the educational aspects of the program: The teaching process and content. The programming aspects are

only of secondary importance.

SOURCES OF INFORMATION

PUBLIC DOMAIN SOFTWARE

Once you have learned the fundamentals of programming, begin studying how other programmers have created their own software. Commercial software is generally "copy protected," so that its programming is hidden from examination except by specialists in hacking--breaking program protection systems. The best place to look is at public domain software--uncopyrighted programs distributed by computer clubs for nominal fees. All the lines of each program can be easily printed for closer examination of programming techniques.

Each of the organizations listed below has developed disks of public domain software for reading and language arts teachers. Each disk contains about ten programs. The software is generally available for simply the cost of the disk and postage. See Figure 14 for a sample listing of some programs.

New Jersey Reading Association Microcomputer Committee
c/o Dept. of Elementary & Secondary Education & Reading,
State University of New York at Geneseo, Geneseo, NY 14454

NJRA Reading/Literature Disk #1

NJRA Reading/Literature Disk #2

NJRA Teacher Utilities Disk #1

Northeast Georgia Council, International Reading Association
Reading/Language Arts Computer Resource Center, 309 Aderhold Hall, University of Georgia, Athens, GA 30602

Set 1--Reading Variety

Set 2--Simulations

Set 3--More Reading Variety

Virginia State Reading Association Microcomputer Committee

Dr. Ronald Magin, 6901 Wild Turkey Drive, Spotsylvania, VA 22553

Reading Language Arts Diskettes #1, #2, #3

CHAN, JULIE M. T., AND MARILYN KOROSTOFF. 1984. Teachers' Guide to Designing Classroom Software. Beverly Hills, Cal.: Sage Publications.

LANDA, RUTH K. 1984. Creating Courseware: A Beginner's Guide. New York: Harper and Row.

Both texts take a step-by-step approach to designing computer-based instructional programs. Emphasis is on program design and the layout of frames. There is no actual programming involved in either text.

BELL, FREDERICK H. 1984. Apple Programming for Teaching and Learning. Reston, Va.: Reston Publishing.

Most programming textbooks use business and mathematical programming as examples and exercises. This textbook uses educational programming to teach principles and commands. A variety of programs are listed, including simple quizzes, drills, and games. The programs are designed for the Apple II-series microcomputers.

REFERENCES

Balajthy, Ernest. 1986. Microcomputers in Reading and Language Arts. Englewood Cliffs, N. J.: Prentice-Hall.

Chan, Julie M. T., and Marilyn Korostoff. 1984. Teachers' Guide to Designing Classroom Software. Beverly Hills, Cal.: Sage Publications.

Crowder, Norman A. 1959. "Automatic Tutoring by Means of Intrinsic Programming." In Automatic Teaching: The State of the Art, ed. Eugene Galanter, pp. 109-116. New York: John Wiley.

Landa, Ruth K. 1984. Creating Courseware: A Beginner's Guide. New York: Harper and Row.

Morris, John M. 1984. "The Case for CAI." Sigcue Bulletin 18, no. 1 (Winter): 11-14.

O'Shea, Tim, and John Self. 1983. Learning and Teaching with Computers. Englewood Cliffs, N.J.: Prentice-Hall.

Rosenshine, Barak, and Robert Stevens. 1986. "Teaching Functions." In Merlin C. Wittrock (ed.), Handbook of Research on Teaching, 3rd ed., pp. 376-391. New York: Macmillan.

Skinner, B. F. 1968. The Technology of Teaching. New York: Appleton-Century-Crofts.

Thorndike, Edward L. 1898. "Animal Intelligence: An Experimental Study of the Associative Processes in Animals." Psychological Review Series of Monograph Supplements 2, no. 4 (June).

Figure -1. Screen page from Quizit

```
BECOME ADJUSTED

THE LENS OF THE EYE WILL ^^^^^^^^^^^
ITS SHAPE FOR DIFFERENT DISTANCES.

-----------------------------------------
WELL. THIS IS TOUGH. PAUL.

THIS WORD HAS 11 LETTERS.

HOW MANY DO YOU WANT (0 TO 11)?
-----------------------------------------
```

Figure 2. Completed Plan Sheet # 1.

TOPIC: Context Analysis

GOAL STATEMENT: Students will learn to recognize and use different types of context clues to determine meanings of unfamiliar words.

GENERAL DIAGRAMMATIC PLAN:

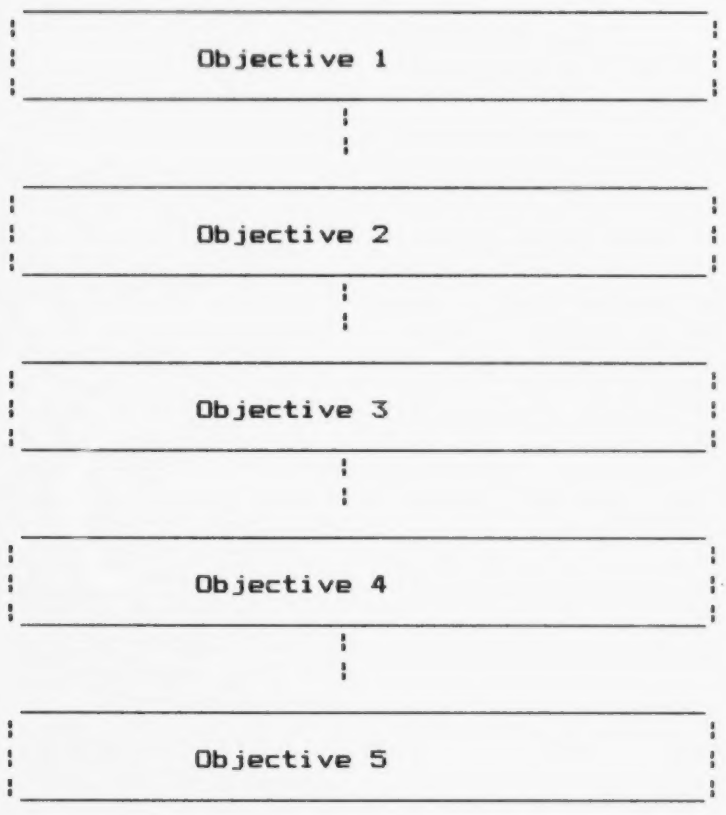

Figure 3: Blank Plan Sheet # 1

NAME: _____

TOPIC:

GOAL STATEMENT:

GENERAL DIAGRAMMATIC PLAN: (Note: You may include more or fewer objectives, as desired.)

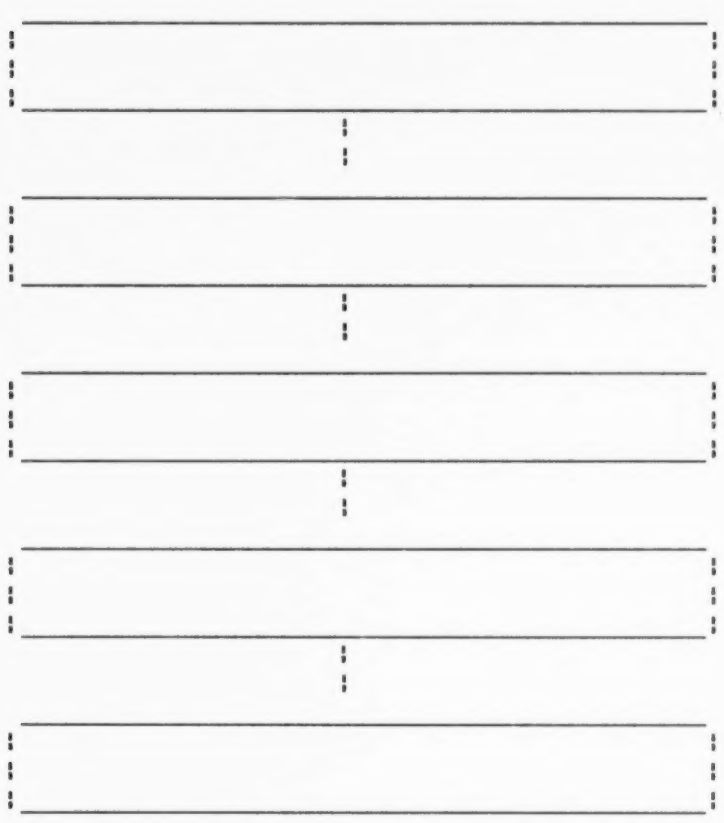

Figure 4. Blank Plan Sheet # 2

Name_____

SPECIFIC OBJECTIVES: (Note: You may include more or fewer objectives, as desired.)

OBJECTIVE 1:

OBJECTIVE 2:

OBJECTIVE 3:

OBJECTIVE 4:

OBJECTIVE 5:

Figure 5. Lesson Plan Sheet # 3

NAME_____

LESSON SUMMARIES: Context Analysis Program

LESSON 1: The term "context" will be defined, and some examples of context--social, physical, and verbal--will be offered. Then the relationship of contextual analysis skills to other word attack skills will be discussed. Students will be shown how contextual analysis strategies can supplement their phonics, structural analysis, and dictionary skills when they encounter unknown words.

LESSON 2: An important realization which students must make is that contextual clues to meaning can vary dramatically in exactness. In some few cases, the exact meaning of an unfamiliar word may be obtainable from the context. In most cases, however, it is possible to obtain only an approximate meaning of the target word--to take a guess at the meaning, as it were. In some other cases, the meaning may be impossible to come by from context clues, or the clues may even suggest a misleading definition.

LESSON 3: This lesson describes the use of antonyms, words of opposite meaning, closely linked to the unfamiliar word as clues to meaning. An example sentence will be given, followed by a detailed explanation of how the antonyms can be understood to be clues to the meaning of a target word. Other examples will be given, requiring student identification of the antonym clues.

LESSON 4: This lesson describes the use of synonyms, words of similar meaning, closely linked to the unfamiliar word as clues

to meaning. An example sentence will be given, followed by a detailed explanation of how the synonyms can be understood to be clues to the meaning of a target word. Other examples will be given, requiring student identification of the antonym clues.

LESSON 5: In many cases, especially in content area texts, the author's purpose is to introduce new terms so that his readers learn them. In such instances the author may use a number of methods to explicitly define the word in context. One way is to use dashes (--) or commas to enclose a definition. Another way is to use phrases which essentially mean "Here is a definition": In other words, that is, and or. This lesson explains and gives examples of several such clues, requiring student identification of each.

Figure 6. Lesson Plan Sheet # 3

NAME_____

LESSON SUMMARIES:

LESSON 1:

LESSON 2:

LESSON 3:

LESSON 4:

LESSON 5:

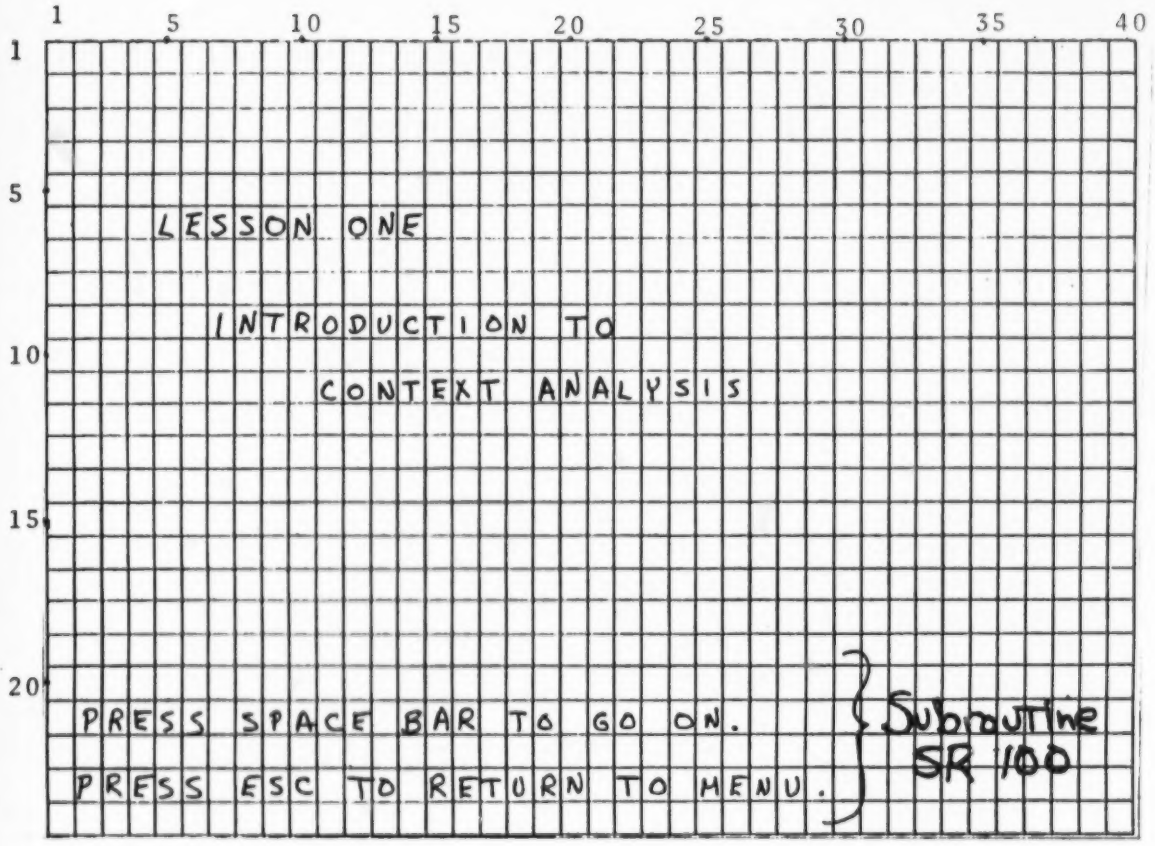

Figure 7 Screen Page #1.

Figure 8 Screen Page #2.

If first response is A, B, or C:

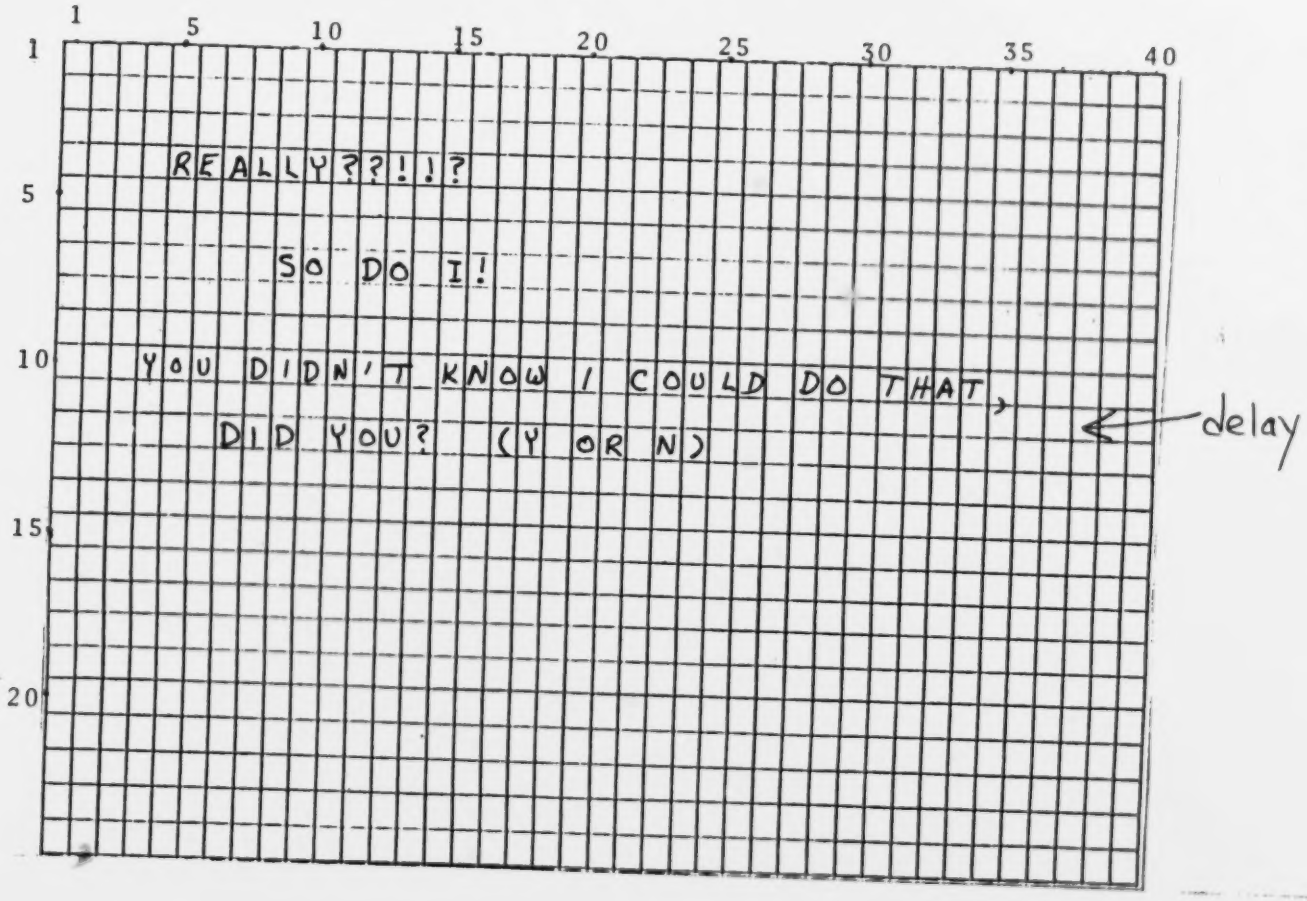

Figure 9. Screen Page #3.

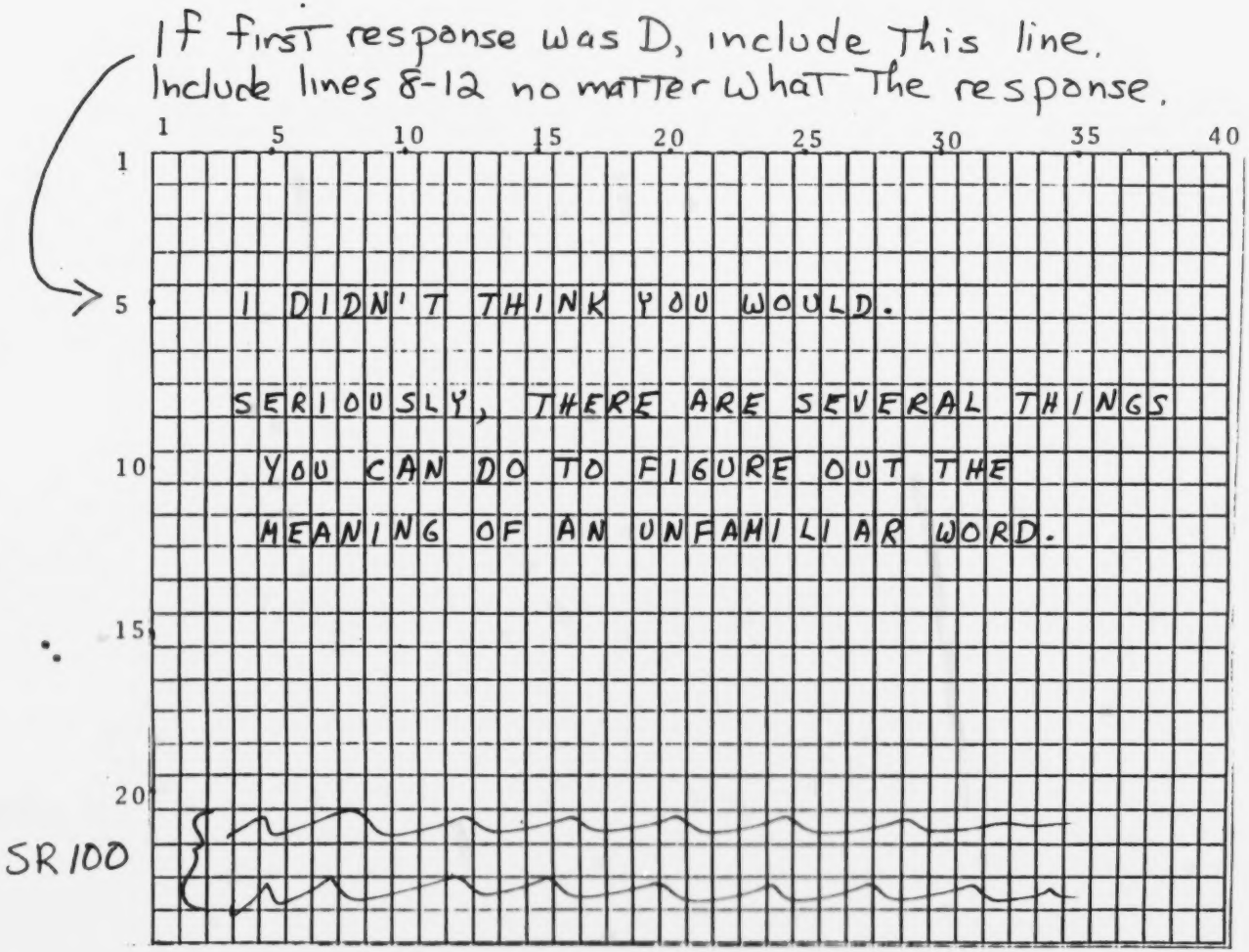

Figure 10 Screen Page #4.

Figure 11 Blank Screen Page. (Apple II-series)

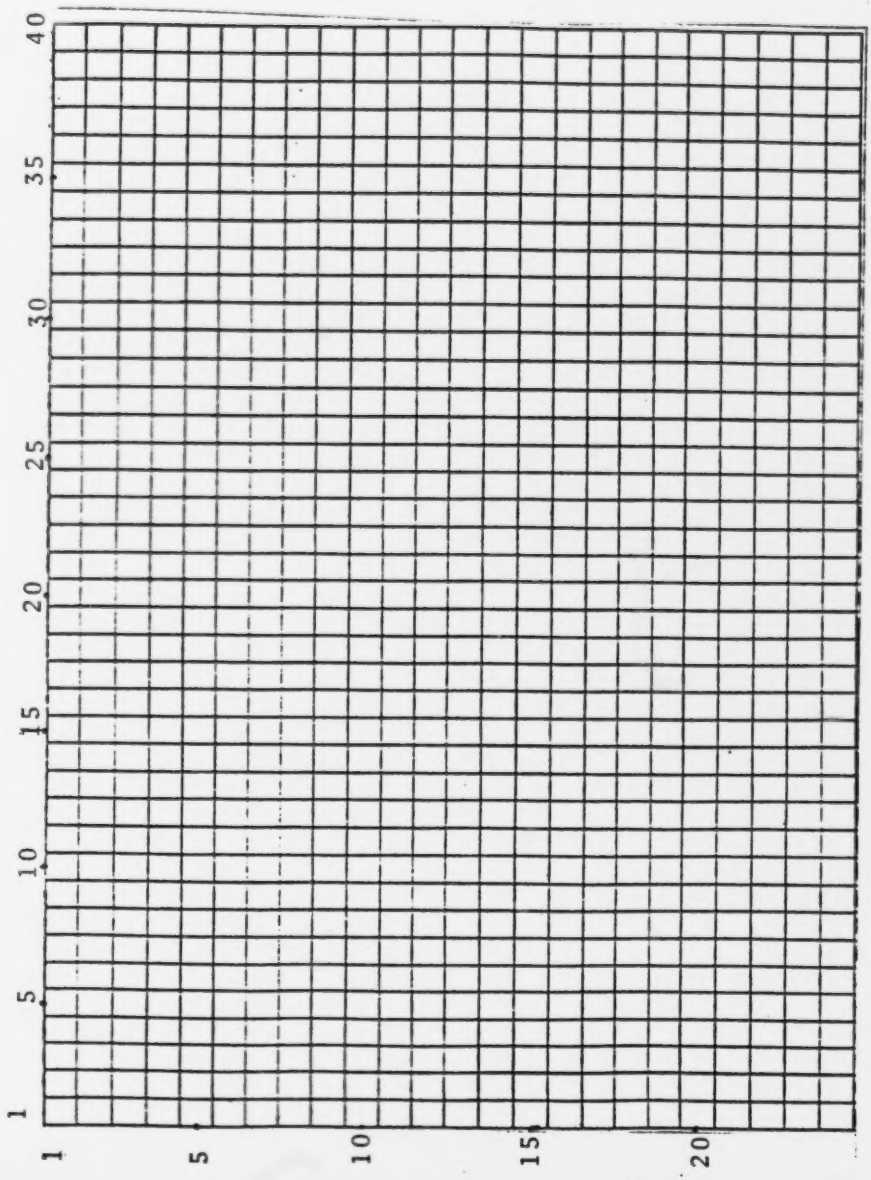

Figure 12. Listing of Lesson One.

```
1  REM  LESSON NUMBER ONE--INTRODUCTION TO CONTEXT ANALYSIS
2  REM  ERNEST BALAJTHY
3  REM  VERSION 9/27/84
4  GOTO 1000
100  REM  PRESS SPACE BAR ROUTINE
105  VTAB 21: PRINT " PRESS SPACE BAR TO GO ON."
110  GET A$
115  RETURN
1000  REM  SCREEN PAGE #1
1005  VTAB (6): HTAB (5): HOME
1010  PRINT "LESSON ONE"
1015  PRINT : PRINT
1020  PRINT "       INTRODUCTION TO"
1025  PRINT : PRINT "           CONTEXT ANALYSIS"
1030  GOSUB 100
1100  REM  SCREEN PAGE #2
1105  VTAB (3): HTAB (4): HOME
1110  PRINT "WHAT DO YOU DO WHEN YOU COME"
1115  PRINT : PRINT "       ACROSS A WORD YOU DON'T"
1120  PRINT : PRINT "       UNDERSTAND?"
1123  FOR WT = 1 TO 3000: NEXT WT: REM    THIS INSERTS A DELAY
1125  VTAB (11): HTAB (5): PRINT "A. CRY"
1130  PRINT : HTAB (5): PRINT "B. THROW A TANTRUM"
1135  PRINT : HTAB (5): PRINT "C. HOLD YOUR BREATH TILL YOU TURN"
1140  HTAB (10): PRINT "BLUE."
1145  PRINT : HTAB (5): PRINT "D. NONE OF ABOVE"
1150  VTAB (21): HTAB (3): INPUT "CHOOSE ONE OF THE ABOVE.  ":A$
1155  IF A$ = "D" GOTO 1300
1200  REM  SCREEN PAGE #3
1205  HOME : VTAB (4): HTAB (5): PRINT "REALLY??!!?"
1210  PRINT : PRINT : VTAB (9): PRINT "SO DO I!"
1215  FOR WT = 1 TO 3000: NEXT WT
1220  VTAB (10): PRINT "   YOU DIDN'T KNOW I COULD DO THAT."
1225  PRINT : PRINT "       DID YOU?  (Y OR N)"
1230  GET A$
1235  GOTO 1315
1300  REM  SCREEN PAGE #4
1305  HOME : VTAB 5: PRINT "   I DIDN'T THINK YOU WOULD."
1315  GOSUB 100
1400  REM  SCREEN PAGE #5
1405  HOME : VTAB (8): PRINT "   SERIOUSLY. THERE ARE SEVERAL THINGS"
1410  PRINT : PRINT "    YOU CAN DO TO FIGURE OUT THE"
1415  PRINT : PRINT "    MEANING OF AN UNFAMILIAR WORD."
1420  GOSUB 100
```

```
1500  REM  SCREEN PAGE #6
1505  HOME : VTAB (4): PRINT "     YOU CAN LOOK"
1510  PRINT : PRINT : HTAB (13): PRINT "INSIDE"
1515  PRINT : PRINT : HTAB (17): PRINT "THE WORD."
1520  GOSUB 100
1525  VTAB 11: PRINT "     THAT IS."
1530  PRINT : PRINT "        YOU CAN LOOK AT ITS PARTS."
1540  PRINT : PRINT "        AT  1) PREFIXES"
1545  PRINT : PRINT "            2) SUFFIXES"
1550  PRINT : PRINT "            3) ROOT WORDS"
1555  GOSUB 100
1600  REM  SCREEN PAGE
1605  HOME : VTAB (4): PRINT "     YOU CAN LOOK"
1610  PRINT : VTAB (6): PRINT "       OUTSIDE"
1615  PRINT : VTAB (8): PRINT "           THE WORD."
1620  VTAB (11): PRINT "     THAT IS."
1625  PRINT : PRINT "        YOU CAN LOOK AT THE WORDS AND"
1630  PRINT : PRINT "          SENTENCES SURROUNDING IT."
1635  GOSUB 100
1700  REM  SCREEN PAGE
1705  HOME : VTAB (4): HTAB (5)
1710  PRINT "YOU CAN LOOK"
1715  PRINT : VTAB (6): PRINT "           THE WORD UP"
1720  PRINT : VTAB (8): PRINT "           IN A DICTIONARY."
1725  GOSUB 100
1800  REM  SCREEN PAGE
1805  HOME : VTAB (4): PRINT "     THESE LESSONS WILL TEACH YOU"
1810  PRINT : PRINT "        ABOUT THE SECOND WAY TO FIND"
1815  PRINT : PRINT "        THE MEANING OF AN UNFAMILIAR"
1820  PRINT : PRINT "        WORD:"
1825  PRINT : HTAB (16): PRINT "CONTEXT"
1830  GOSUB 100
1900  REM  SCREEN PAGE
1905  HOME : VTAB 4: PRINT "     CONTEXT ANALYSIS MEANS"
1910  PRINT : PRINT "       USING WORDS. PHRASES. AND"
1915  PRINT : PRINT "       SENTENCES SURROUNDING A WORD"
1920  PRINT : PRINT "       TO DETERMINE ITS MEANING."
1925  GOSUB 100
2000  REM  SCREEN PAGE
2005  HOME : VTAB 8: PRINT "    TYPE THE WORD 'CONTEXT' BELOW."
2010  VTAB 13: HTAB 7: INPUT A$
2015  IF A$ = "CONTEXT" THEN  VTAB 15: HTAB 15: PRINT "GOOD!"
2020  IF A$ <  > "CONTEXT" GOTO 2005
9000  PRINT  CHR$ (4):"RUN LESSON TWO"
```

Figure 13 Listing of Lesson Two.

```
1  REM   LESSON NUMBER TWO--
2  REM  ERNEST BALAJTHY
3  REM  VERSION 9/27/84
4  GOTO 1000
100  REM  PRESS SPACE BAR ROUTINE
105  VTAB 21: PRINT " PRESS SPACE BAR TO GO ON."
110  GET A$
115  RETURN
1000  REM  SCREEN PAGE #1
1005  VTAB (6): HTAB (5): HOME
1010  PRINT "LESSON TWO"
1015  PRINT : PRINT
1020  PRINT "     HOW MUCH"
1025  PRINT : PRINT "            CAN CONTEXT HELP? "
1030  GOSUB 100
1100  REM  SCREEN PAGE #2
1105  HOME : VTAB (4): PRINT "    CONTEXT CAN HELP YOU GUESS"
1110  PRINT : PRINT "       THE MEANING OF WORDS."
1115  PRINT : PRINT : PRINT "    SOMETIMES YOU WILL BE ABLE"
1120  PRINT : PRINT "       TO GUESS THE APPROXIMATE"
1125  PRINT : PRINT "       MEANING."
1130  PRINT : PRINT : PRINT "    LOOK AT THE FOLLOWING EXAMPLE."
1135  GOSUB 100
1200  REM  **********
1205  HOME : VTAB (4): PRINT "    MANY DIFFERENT KINDS OF BIRDS"
1210  PRINT : PRINT "      CAN BE SEEN IN THE ZOO:  CROWS"
1215  PRINT : PRINT "      CURLEWS. BLUEBIRDS. AND FINCHES."
1220  VTAB 13: PRINT "    FROM THE CONTEXT. CAN YOU FIND"
1225  VTAB 15: PRINT "      ANY INFORMATION ABOUT WHAT A"
1230  VTAB 17: INPUT "      CURLEW IS?  (Y OR N)":A$
1235  IF A$ = "N" THEN  VTAB 19: PRINT "SURE YOU CAN!": FOR WT = 1 TO 300
    0: NEXT WT
1240  VTAB 13: PRINT "    WHAT IS A CURLEW?                    "
1250  VTAB 15: PRINT "      A CURLEW IS A __  .                "
1253  VTAB 17: PRINT "                                        "
1255  VTAB 15: HTAB 21: INPUT A$
1260  IF A$ = "BIRD" THEN  GOTO 1280
1265  VTAB 19: HTAB 25: PRINT "NO."
1268  FOR WT = 1 TO 3000: NEXT WT
1270  VTAB 15: HTAB 21: PRINT "BIRD"
1275  GOSUB 100
1280  VTAB 19: PRINT "                              GOOD."
1285  GOSUB 100
1300  VTAB 13: PRINT "    THE SENTENCE ABOVE GIVES ONLY      "
1305  VTAB 15: PRINT "       APPROXIMATE CONTEXT.            "
1310  VTAB 17: PRINT "    THAT IS. WE HAVE A GENERAL IDEA AS"
1315  VTAB 19: PRINT "    TO MEANING. BUT NOT EXACT.          "
1320  GOSUB 100
1400  REM  SCREENPAGE
1405  HOME : VTAB 3: PRINT "   SOMETIMES CONTEXT CAN ACTUALLY "
1410  PRINT : PRINT "     MISLEAD A READER."
1500  REM  **********
```

```
1505  HOME : VTAB 3: PRINT "   SOMETIMES CONTEXT CAN ACTUALLY"
1510  PRINT : PRINT "     MISLEAD A READER"
1515  VTAB 8: PRINT "   IN THESE CASES. THE CONTEXT"
1520  VTAB 10: PRINT "    SUGGESTS AN INCORRECT MEANING"
1525  VTAB 12: PRINT "    FOR THE WORD."
1530  VTAB 15: PRINT "   LOOK AT THE NEXT EXAMPLE."
1535  GOSUB 100
1600  REM  **********
1605  HOME : VTAB 4: PRINT "    THE CURLEWS WALKED ALONG THE"
1610  VTAB 6: PRINT "      PIER. GAZING AT THE FISHERMEN"
1615  VTAB 8: PRINT "      AND HUNGRILY WISHING FOR DINNER."
1620  VTAB 11: PRINT "  FROM THIS CONTEXT. WHAT MIGHT A "
1625  VTAB 13: PRINT "    CURLEW BE?"
1630  VTAB 15: INPUT "     COULD IT BE A PERSON? (Y OR N)  ":A$
1635  IF A$ = "Y" THEN  VTAB 18: PRINT "         YES. YOU'RE RIGHT!"
1640  GOSUB 100
1645  VTAB 18: PRINT "                                        "
1650  VTAB 15: INPUT "     COULD IT BE A BIRD? (Y OR N)      ":A$
1655  IF A$ = "Y" THEN  VTAB 18: PRINT "          YES. YOU'RE RIGHT!"
1660  GOSUB 100
1665  VTAB 18: PRINT "                                        "
1670  VTAB 15: INPUT "     COULD IT BE A CAT? (Y OR N)       ":A$
1675  IF A$ = "Y" THEN  VTAB 18: PRINT "          YES. YOU'RE RIGHT!"
1680  GOSUB 100
1685  VTAB 18: PRINT "                                        "
1690  VTAB 11: PRINT "ACTUALLY. FROM THE CONTEXT ABOVE."
1695  VTAB 13: PRINT "    A CURLEW COULD BE ANY OF THOSE"
1700  VTAB 15: PRINT "    THREE GUESSES.                      "
1705  VTAB 17: PRINT "   WHAT DO WE KNOW ABOUT CURLEWS FROM"
1710  VTAB 19: PRINT "    THIS SENTENCE?"
1715  GOSUB 100
1720  VTAB 11: PRINT "  CURLEWS...                            "
1723  FOR WT = 1 TO 2000: NEXT WT
1725  VTAB 13: PRINT "  1) WALK                               "
1730  FOR WT = 1 TO 2000: NEXT WT
1735  VTAB 13: HTAB 18: PRINT "2) CAN BE ON A PIER"
1740  FOR WT = 1 TO 2000: NEXT WT
1745  VTAB 15: PRINT "  3) HAVE EYES                          "
1750  FOR WT = 1 TO 2000: NEXT WT
1755  VTAB 17: PRINT "  4) EAT (FISH. MOST LIKELY. BUT THEY   "
1760  VTAB 19: PRINT "       MIGHT ALSO ENJOY FISHERMEN.)     "
1765  GOSUB 100
1770  VTAB 11: PRINT "  WE COULD VERY LOGICALLY CONCLUDE"
1775  VTAB 13: PRINT "   THAT CURLEWS MIGHT BE                "
1780  VTAB 15: PRINT "     PEOPLE. OR CATS. OR BIRDS.         ": PRINT
1785  VTAB 17: PRINT "  SO WE SEE THAT CONTEXT CLUES CAN     "
1790  VTAB 19: PRINT "  MISLEAD US INTO MAKING WRONG GUESSES."
1795  GOSUB 100
1800  REM  **********
```

```
1805  HOME : VTAB 4: PRINT "   IN SOME CASES. CONTEXT CLUES LET"
1810  VTAB 6: PRINT "     US KNOW QUITE EXACTLY THE MEANING"
1815  VTAB 8: PRINT "     OF AN UNFAMILIAR WORD."
1820  VTAB 11: PRINT "   LOOK AT THIS NEXT EXAMPLE."
1825  GOSUB 100
1830  HOME : GOSUB 2000
1835  VTAB 13: PRINT "  WHAT COLOR ARE CURLEWS?"
1840  VTAB 15: HTAB 11: PRINT "_    "
1845  VTAB 15: HTAB 10: INPUT A$
1850  IF A$ = "GREY" THEN  VTAB 17: HTAB 26: PRINT "GOOD!": GOTO 1860
1855  VTAB 15: HTAB 11: PRINT "GREY"
1860  GOSUB 100
1865  HOME : GOSUB 2000
1870  VTAB 13: PRINT "  ARE THE BILLS OF CURLEWS LONG OR"
1875  VTAB 15: PRINT "   SHORT?"
1880  VTAB 17: HTAB 11: PRINT "____"
1885  VTAB 17: HTAB 10: INPUT A$
1890  IF A$ = "LONG" THEN  VTAB 17: HTAB 26: PRINT "GOOD!": GOTO 1900
1895  VTAB 17: HTAB 11: PRINT "LONG   "
1900  GOSUB 100
1905  HOME : GOSUB 2000
1910  VTAB 13: PRINT "  WHAT DO CURLEWS EAT?"
1915  VTAB 15: HTAB 11: PRINT "___  "
1920  VTAB 15: HTAB 10: INPUT A$
1925  IF A$ = "FISH" THEN  VTAB 17: HTAB 26: PRINT "GOOD!": GOTO 1935
1930  VTAB 15: HTAB 11: PRINT "FISH"
1935  GOSUB 100
1940  HOME : GOSUB 2000
1945  VTAB 13: PRINT "  WHERE DO CURLEWS LIVE?"
1950  VTAB 15: HTAB 8: PRINT "NEAR THE ____  "
1955  VTAB 15: HTAB 17: INPUT A$
1960  IF A$ = "BEACH" THEN  VTAB 17: HTAB 26: PRINT "GOOD!": GOTO 1985
1965  IF A$ = "SHORE" THEN  VTAB 17: HTAB 26: PRINT "GOOD!": GOTO 1985
1970  IF A$ = "OCEAN" THEN  VTAB 17: HTAB 26: PRINT "GOOD!": GOTO 1985
1980  VTAB 15: HTAB 18: PRINT "SHORE"
1985  GOSUB 100
1990  GOTO 2100
2000  REM  EXACT EXAMPLE
2005  VTAB 2: PRINT "    THE FATHER AND SON LOOKED OUT OVER"
2010  VTAB 4: PRINT "     THE BREAKING WAVES AT THE LONG-"
2015  VTAB 6: PRINT "     BILLED GREY CURLEWS FLYING IN WIDE"
2020  VTAB 8: PRINT "     CIRCLES AND DIPPING DOWN TO CATCH"
2025  VTAB 10: PRINT "     FISH."
2030  RETURN
2100  HOME : VTAB 3: PRINT "  WE HAVE MADE SOME VERY ACCURATE"
2105  VTAB 5: PRINT "   GUESSES ABOUT CURLEWS. BASED ON"
2110  VTAB 7: PRINT "   CONTEXT."
2115  VTAB 10: PRINT "  CURLEWS ARE SHORE BIRDS WHICH EAT"
2120  VTAB 12: PRINT "   FISH.  THEY ARE GREY AND HAVE LONG"
2125  VTAB 14: PRINT "   BILLS."
2128  GOSUB 100
9000  PRINT  CHR$ (4):"RUN LESSON THREE"
```

Figure 14. New Jersey Reading Association Public Domain Software List.

Reading/Language Arts Diskette #1

Opinion Exercise	Alphabet Soup
Favorite Words	Sayings
Pet Pit Pat Pot	Scrambled
Mathspell	Vowel Search
Alphabet Antics	Logic-1, Logic-2, Logic-3
Aphorism	Boggle
Word Flash	Speed Reading
Word Challenge	

Reading/Language Arts Diskette #2

Letter Recognition	Cryptogram
Letter Guessing Game	Poet
Soothsayer	Hangman-2
Alphabet & Sound	

Teacher Utility Diskette #1

Address Labels	Love
Banner	Diamond
Test Scorer	Visiscore
Electronic Mailbox	Grade
Readability Checker	Find-A-Word Create